

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

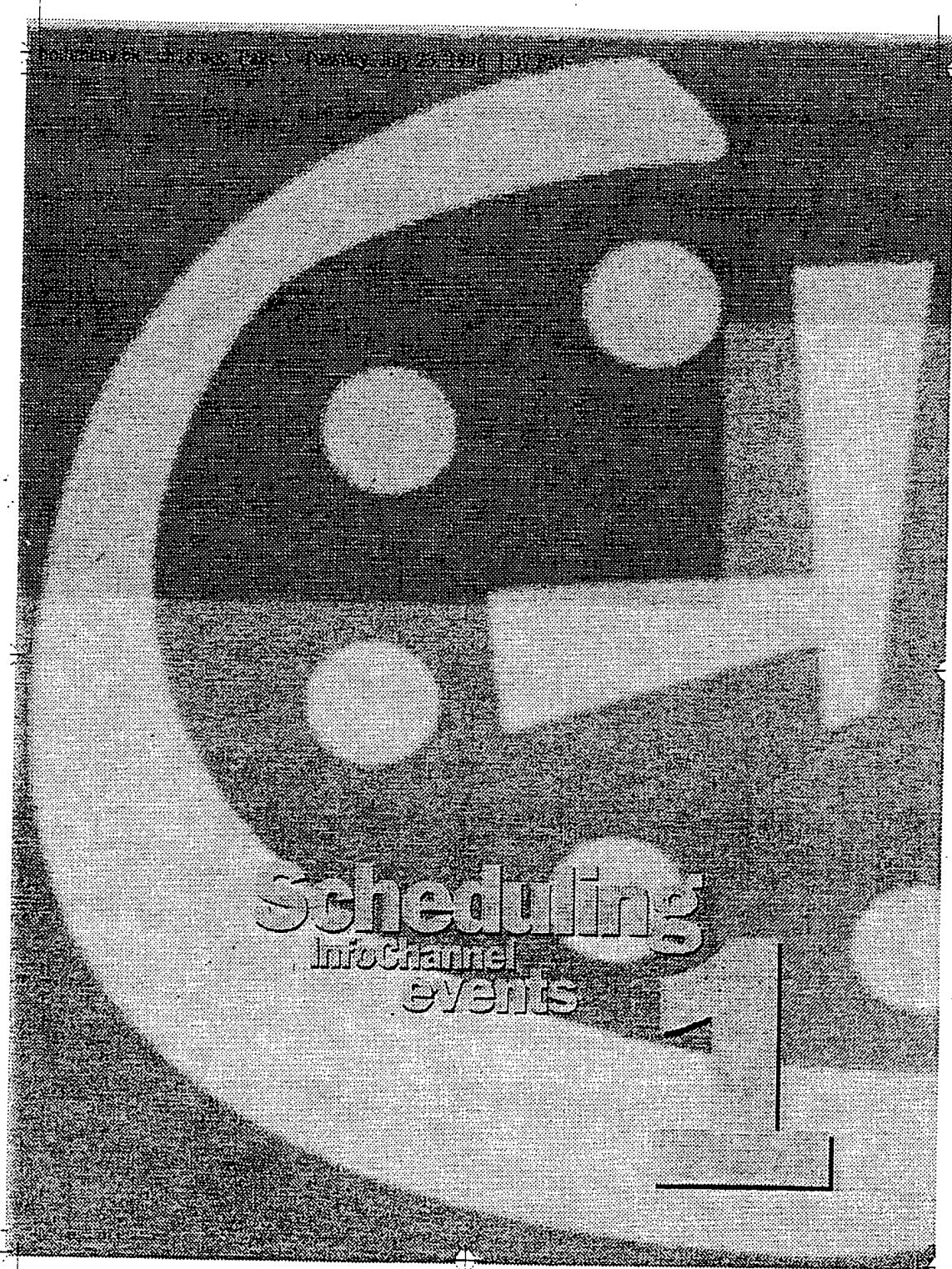
Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**



Scheduling
InfoChannel
events

1: Scheduling InfoChannel events

Scheduling is a key element of many Scala InfoChannel distributed productions, and InfoChannel's scheduling abilities distinguish it from most other multimedia products. Scheduling lets you communicate timely information, freeing you to leave Player Stations unattended even when displayed information must change from day to day.

With InfoChannel, you can schedule events to run according to various criteria:

- at regular intervals—hourly, daily, or weekly
- continuously between two given times or dates
- once, at a specific date and time

For example, you might need an event to be repeated between two and four o'clock in the afternoon, only on Fridays, during January. This kind of scheduling is useful when promoting special events and prevents the display of outdated information. Scheduling a page in your script to appear only once at a specific time can be used, for example, to start CD audio automatically at a certain time.

This chapter discusses scheduling in general and focuses on using the Schedule menu. Before we look at that menu in detail, we first discuss the whys, whats and hows of scheduling.

Scheduling and the Scala InfoChannel Player

InfoChannel lets you communicate continuously. The InfoChannel Player Station is set up to run InfoChannel non-stop, 24 hours a day if the hardware is left on. This capability is what makes scheduling so useful and so important.

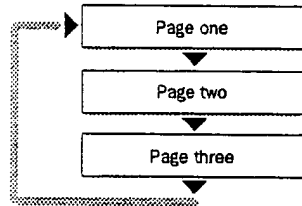
You configure the InfoChannel Player to run just one script (although this script may be updated later—see chapter 2 of this guide, "Using ScalaNet"). That script, the Player Script, runs continuously. Recall that all Scala scripts automatically loop by default, starting over at page

1: Scheduling InfoChannel events

Scheduling and the Scala InfoChannel Player

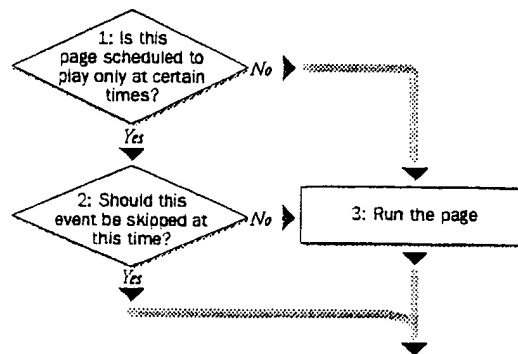
one after the last page is reached. Thus, the simplest InfoChannel production is a script that repeats itself in the same way, day after day.

For example, if an InfoChannel script you create consists of only three pages without scheduling or other advanced features, the Player runs page one, then page two, then page three, then page one, then page two, and so on, ad infinitum.



Productions involving scheduling, however, display different information at different times, or on different days. The Player for these productions still runs a single script containing all events. The script still automatically loops. What is different is that not every event is run on every pass through the script. Some events in the script are skipped over because it is not time to use them.

By default, an event has no schedule, and runs every time that script execution reaches it. When a schedule is set for an event, that event is skipped except as scheduled.



1: Scheduling InfoChannel events

Types of scheduling

What can be scheduled?

You use the Schedule menu for all of the following (although some have limited scheduling options, explained in the next section, "*Types of scheduling*"):

- an *element* on a screen page such as text, clips and sounds
- a *screen page*, consisting of a background and other elements
- a *special event*, which is an event not related directly to the screen display, such as a timing event or a command to an external device like a CD player
- a *group*, a collection of pages that you can treat as a unit
- a *sub-script*, another script that is run from within the current script

Scheduling works essentially the same with screen pages, special events, elements, sub-scripts and groups, and we use the term *events* to apply to all five. Some types of scheduling don't apply to elements, however. Where the distinction is important, we use the term *page* the same way that it is used in the Main menu—to apply to any page, be it a screen page, special event page, group or sub-script, but not to an element of a screen page.



Types of scheduling

There are two types of scheduling in Scala: *periodic* and *exact*. Each has its particular uses, depending on what your production needs are. Periodic scheduling can be applied to any event, while exact scheduling can be applied only to pages (including special event pages, groups and sub-scripts).

Periodic scheduling

Periodic scheduling creates "windows" of time, within which events may run. As Scala runs a script and reaches an event with periodic scheduling set, it checks to see whether the current date and time is

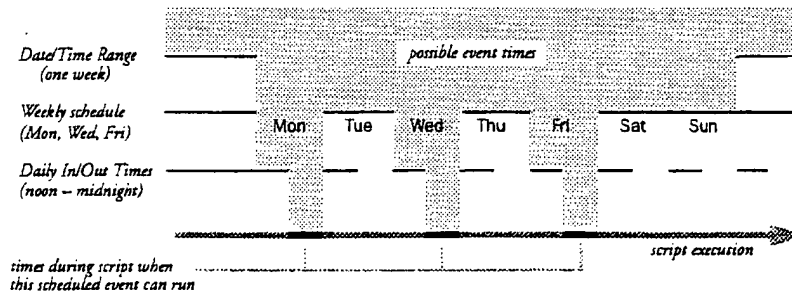
1: Scheduling InfoChannel events

Types of scheduling

between the start and end times defined by the schedule. If it is, the event runs. If not, it is skipped.

It is important to emphasize what periodic scheduling does *not* do: it does not cause events to run at a certain time, or guarantee that an event will run during any given scheduled period. It simply allows the event to run if, in the normal pace and sequence of the script's execution, the event is reached during the scheduled period. In short, periodic scheduling specifies when something *can* happen, not necessarily when (or even if!) it *will* happen.

The following diagram graphically illustrates how the three levels of periodic scheduling work together to "filter" the possible execution times of an event. The example schedule has an overall range of one week (all day each day), with Monday, Wednesday, and Friday selected and daily start and end times of noon and midnight:



Exact scheduling

Exact-scheduled events work differently. During normal running of the script, Scala always skips over exact-scheduled pages. However, when the scheduled time arrives, InfoChannel interrupts whatever else is happening in order to run the page that is exact-scheduled for that time. After running an exact-scheduled page, Scala continues by running the next page in the script after the exact-scheduled page. It does not return to the interrupted page (unless, by coincidence, that hap-

1: Scheduling InfoChannel events

Working in the Schedule menu

pened to be the next page after the exact-scheduled page in the script sequence).

For periodic-scheduled events, the location of the event in the script dictates when the event may run. For exact-scheduled pages, the location of the page in the script is irrelevant to when the event will run. It does, however, dictate what page will run next after the exact-scheduled page.

Many of the important issues in scheduling are related to issues in timing, so you should be familiar with chapter 14, "*Adjusting the timing of events*". In particular, you must keep in mind that the timing of events is relative. Neither type of scheduling adjusts pause lengths, wipe speeds, or other factors that control a production's pacing, in order to cause something to happen at or by a certain time. You must adjust these factors carefully yourself to ensure that events can occur as scheduled.

To make the best use of scheduling in a complex production, you will sometimes need to utilize grouping and/or sub-scripts. To ensure that these tools are at your disposal, you may wish to review chapter 3 of the *Producing* guide, "*Working in the Main menu*", where these topics are introduced. You might also take advantage of branching, explained in chapter 17, "*Advanced techniques*". You should be aware that the use of groups and/or sub-scripts has special considerations (for details, see the chapters just mentioned and the section "*Scheduling with groups and sub-scripts*", later in this chapter).

The explanations and examples included in this chapter should help to clarify the ways in which the different types of scheduling work, and how to choose which to use in your production.

Working in the Schedule menu

To see the *Schedule* column on the Main menu, you may need to use the horizontal scroll bar. When you click the *Schedule* button for a

1: Scheduling InfoChannel events

Working in the Schedule menu

page in the Main menu, or an element in the List menu, you see the Schedule menu.

The Schedule: selects/determines whether an entry is periodic or exact

You use the Schedule menu to specify one or more schedule *entries* in the schedule for the event. *Periodic* entries determine whether the event will happen as the executing script reaches that event. *Exact* entries determine that the executing script is to be interrupted at a particular time to run a particular page.

There are several controls for individual schedule entries, and most can be combined to narrow a schedule in different ways. Combinations give you more power and flexibility in scheduling.

Schedule entries

The schedule for one event can contain more than one entry. You might, for example, want the same page to be run only during an hour in the morning and another hour in the afternoon. Each hour would be one entry in the schedule for that page.

When you open the Schedule menu, you start at entry 1. Once you've edited it to your satisfaction, you can add new entries. To add another entry, click the *Add Entry* button. To edit or review other entries in the schedule for a page once you have more than one entry, use the *Entry Number* value control to move to the different entries.

Moving to a different entry, using either the *Add Entry* button or the *Entry Number* control, is like clicking *OK* for the previous entry: it

1: Scheduling InfoChannel events

Working in the Schedule menu

automatically confirms any changes to the current entry starting with the next entry.

The schedule for an event is the combination of all the schedule entries for the event. Schedule entries are "affirmative": they specify when an event *can* run. You can't schedule an event *not* to run, except by not having any schedule entry for that time. Thus, schedule entries generally cannot override each other. (Nonsensical schedules can be created such that several schedule entries for an event ensure that it always runs, or such that exact-scheduled events interrupt each other).

You can delete a single schedule entry (the current one) with the *Remove Entry* button, and delete all the entries in the schedule for that event with the *Remove All* button. With either *Remove* button, you are asked to confirm, because you cannot later undelete or cancel to recover deleted schedule entries.

You can edit the schedule for only a single event at a time. To edit the schedule for another event, exit the Schedule menu and click the *Schedule* button for the event you wish to edit. You cannot multi-select several events and set the schedule for all of them at the same time, but you can group multiple pages to give them all the same schedule.

To accept the changes you have made to the displayed schedule entry and exit the Schedule menu, click *OK*. To discard changes and exit the menu, click *Cancel*.

Defaults

In the Schedule menu, times are displayed in 24-hour hours:minutes:seconds format, and default start and end times are 00:00:00 and 23:59:59, respectively. Thus the default Start time or In time is at the beginning of the day (at midnight) and the default End time or Out time is at the end of the day (one second before midnight).

Unless a schedule entry is explicitly made *exact* by changing the *Schedule* selector from *Active period* to *Interrupts exactly as*, it is periodic. Most of the Schedule menu's controls apply whether the entry is *exact* or periodic.

1: Scheduling InfoChannel events

Working in the Schedule menu

The Schedule Date/Time Range controls

The *Date* and *Time* controls to the right of the *Start?* and *End?*

InfoChannel Tip

It might be easier to think of the *Schedule Date/Time Range* controls as specifying the window outside of which the event cannot possibly run.

buttons allow you to set the overall period within which an event can run. When you turn either *Start?* or *End?* on (✓), its *Date* and *Time* value controls become available. If you adjust only a date, the default times are used. If you need to be more specific, use the *Time* value controls to specify the beginning and ending points of the range more precisely.

These controls are the most general of the scheduling controls. You further restrict the days and times within which the event can run with the *Daily* and *Weekly* controls. Setting an End date for an event without setting a start date implies a start date of *now*; setting a Start date without an End date implies that the schedule continues indefinitely.

Example 1: Ten days of promotional pricing

Suppose your Scala production runs on displays in a chain of 24-hour gas stations. One of the pages in the production extols the virtues of a premium grade gasoline. In February 1996, you are told that you should get ready to promote special promotional pricing on that page, for a sale which is valid only from March 1, 1996 at 8 AM through March 10, 1996 at midnight.

Here is how you would create this:

1. Create a new element on the premium gasoline page, a clip which wipes the price and expiration date onto the page.
2. Click the *Schedule* button for the clip to open the Schedule menu.
3. Set a start date of March 1, 1996, by clicking *Start?* (✓), and then adjusting the *Date* value control.
4. Set a start time of 08:00:00 by using the *Time* value control.
5. Set an end date of March 10, 1996 by clicking *End?* (✓) and then adjusting its *Date* value control.

1: Scheduling InfoChannel events

Working in the Schedule menu

6. Because you want the promotion to run all day on the 10th, you don't have to specify an end time. You finish editing the scheduling by clicking *OK*.

After 8 AM on March 1, and until the end of March 10, every time your script reaches the promotional pricing event, the clip wipes in on the page. At any other time, it doesn't run.

Once your new script is on the gas station Players, you won't have to do anything more to get the display of sale information started, or to stop it. You won't have to delete the scheduled element from the page to keep it from appearing once the sale ends. You might choose to delete it later to keep the script small, or you might choose to keep it there in case you are asked to schedule it again.

The Daily In/Out Times controls

To run an event only during a certain time period each day, you adjust the *In Time* and *Out Time* value controls to specify the time of day to begin and end running the event. You may run the event for the rest of the day (stopping at midnight), or adjust the *Out Time* value control to specify when to stop running the event until the *In Time* for the next day. These daily times apply to every day that is permitted by the *Weekly* and *Date/Time Range* settings. If the *Out* time occurs after the event has already started, the event completes; it is not cut off.

Keep in mind that if your script is long and the *In/Out* time is short, your event could be skipped altogether—the running script might not reach the event during the specified time. As well, with a short script, your event may be run many times. With periodic scheduling, there is no way of specifying how many times an event is to run within the time “window” specified by the schedule.

Example 2: the lunch page

For this example, suppose your Scala production is an internal corporate information channel. You notice the performance of too many employees sagging during afternoons when they've skipped lunch. So you decide to add to your production a humorous noon-hour reminder that it's time to get out of the office for a few minutes and get something to eat. After creating your new lunch page, you would:

1: Scheduling InfoChannel events

Working in the Schedule menu

1. Go to the Schedule menu for the lunch page.
2. Adjust the *In Time* value control to set an In time of 12:00:00.
3. Adjust the *Out Time* time value control to set an Out time of 13:00:00.
4. Click *OK* to accept your changes to the schedule for this page.

Now, whenever the script gets to your page during the noon hour, it runs. During the rest of the day, your lunch page is skipped.

Weekly

At the bottom of the Schedule menu are buttons which may be used to limit the days of the week that an event runs. If none of them is selected, the event runs the every day of the week (within the limits, if any, of the *Schedule Date/Time Range* controls). If you select a weekday button, you limit the event to running only on that day. You can select any combination of days during the week.



the Weekly setting for an event that runs only on weekdays

Example 3: Today's menu

With a Scala production running on the hotel information channel, you might want to promote the hotel restaurant's dinner specials. Since there is a special for each day of the week that stays the same week after week, you might prepare a page for each, all bearing the text "Today's Menu". (You could even create a page for each course, and group the pages into menus for each day of the week! Remember, a group is scheduled like any other page.) After creating the pages, scheduling each of them is simple — you go to that page's Schedule menu and click the day on which the page is to be displayed. Each page runs only on the selected day, with the pages for unselected days being skipped.

Later, your chef might decide that seven menus is too many and that he won't be cooking duck on Thursdays any more. Instead, he will

1: Scheduling InfoChannel events

Working in the Schedule menu

serve the same thing on Tuesdays and Thursdays. No problem. After you delete Thursday's menu page, go to the Schedule menu for Tuesday's page. Click on the *Thu* button to add Thursday to the schedule (so that both the *Tue* and the *Thu* buttons are selected), and then the page runs on both Tuesdays and Thursdays.

Finally, if your menus are seasonal, you could create four sets of menus: winter, spring, summer and fall. You would use appropriate Start and End dates to automatically change displayed menus with the seasons as well as by day of the week. You would need to update schedule entries each year.

Scheduling pages at an exact time

Exact scheduling and periodic scheduling are not mutually exclusive. In fact, when using exact scheduling, the *Schedule Date/Time Range* and *Weekly* controls function in the same way as they do with periodic scheduling. Switching to exact scheduling only changes the way the *Daily In/Out Times* controls work. Remember, however, that elements on a page, unlike other events, cannot be exact-scheduled. Exact scheduling is available only for pages.

To exact-schedule a page, change the *Schedule:* selector from *Active period* to *Interrupt exactly at*. The *In Time* value control now indicates the first (or only) time you wish the page to run. You see two new buttons, *Repeat?* and *Interval*, for exact scheduling. The *Out Time* control remains, but is disabled unless you are using repeats.

available only with
exact scheduling



Repeat and Interval

Without using any other controls, your exact-scheduled page will run once per day at the time specified by *In Time*. You might prefer to repeat the page, once every hour, for example. To do so, turn on *Repeat?* (✓). This enables the *Interval* value control, so you can specify how often to repeat the page. The time interval you specify, in hours, minutes, and seconds, is measured from the last time the page began.

1: Scheduling InfoChannel events

Working in the Schedule menu

You can then use *Out Time* to set a time to stop repeating the page, or leave the Out time at 23:59:59 to repeat the page for the rest of the day.

The other controls in the Schedule menu (such as *Schedule Date/Time Range* and *Weekly*) can be used to further limit when an exact-scheduled event will occur.

Remember, periodic-scheduled events run only when script execution reaches them. But when an exact-scheduled page's time comes, it interrupts whatever the script is doing at that moment. It will interrupt in the middle of an element wipe just as readily as in the middle of a page pause.

After an exact-scheduled page runs (whether the first time or a repeated time), the script continues normally from the page that immediately follows the exact-scheduled page in the script, until the time for the next exact-scheduled page or exact-scheduled repeat.

Example 4: The \$1,000 prize

To reward your loyal viewers, you might have a cash-prize giveaway on your cable TV barker channel. When a special message is displayed, the first viewer to call in wins. To display the message, design a page for your production giving the details. Pick the date and time you want the page displayed, and set an exact schedule for the page at that date/time. To do so:

1. Click *Start?* (✓) and set its *Date* value control.
2. Click *End?* (✓) and set its *Date* value control to match the start date.
3. Click *In Time?* (✓) and set its value control.
4. Switch to the *Schedule:* selector to *Interrupt exactly at*.

Your page will be displayed only once, on the date and at the time you specified, and it will interrupt whatever the barker channel script was doing at exactly that time.

1: Scheduling InfoChannel events

Scheduling with groups and sub-scripts

View

View...

With the **View** menu (accessible from the *View* button), you can look at the schedule entries for the current event or for the whole script. If your scheduling is complex, this is a useful and important way to review your schedule.



This illustration shows both a periodic and an exact event, with all the basic schedule information, including the repeat interval for the second page.

Scheduling with groups and sub-scripts

If your scripts use groups or sub-scripts, you should be aware of the possible effects on the execution of scheduled events.

Exact scheduling works regardless of whether script execution is currently within a group or sub-script. Exact-scheduled pages cannot be created or moved to the inside of a group, so grouping has no effect on their execution. Although exact-scheduled pages can exist in sub-scripts, exact-scheduled pages in sub-scripts also are not affected.

So if the start time of an exact-scheduled page in the main script occurs while a sub-script or grouped page is running, InfoChannel returns to the main script to run the exact-scheduled page. If the exact-scheduled page is in a sub-script, InfoChannel exits the current main script or sub-script and the exact-scheduled page runs. In any case,

1: Scheduling InfoChannel events

Scheduling with groups and sub-scripts

after an exact-scheduled page runs, the script continues not with the interrupted page, but with the page that immediately follows the exact-scheduled page in its script.

Periodic-scheduled events, on the other hand, may not execute according to schedule when groups or sub-scripts are involved. A periodic event runs only when it is actually reached during normal flow of the script. Thus, as the script steps through all events, including those in groups and sub-scripts, if it reaches the periodic-scheduled event within a "window" of time specified by the event's schedule, the event runs. But if a sub-script or group uses branching to create a loop such that the main script is never returned to, periodic-scheduled events in the main script never run, just as non-scheduled events in the main script won't run.

How not to schedule

While the scheduling features of InfoChannel are very powerful, they are not without limitation.

In Scala, events take a variable amount of time, depending on hardware and other considerations (see chapter 14, page 304, "*Factors that affect timing*"). Thus you cannot reasonably set a schedule in InfoChannel of what is going to be displayed during each minute of the day. You must think about your scheduling needs in a different way. The following example illustrates some potential problems.

Example 5: The lunch page gone wrong

A previous example showed a good way to display a lunch page during the noon hour. But by neglecting to think of how events have relative timing and not considering where a script continues after being interrupted by an exact-scheduled page, our lunch page might have turned out differently.

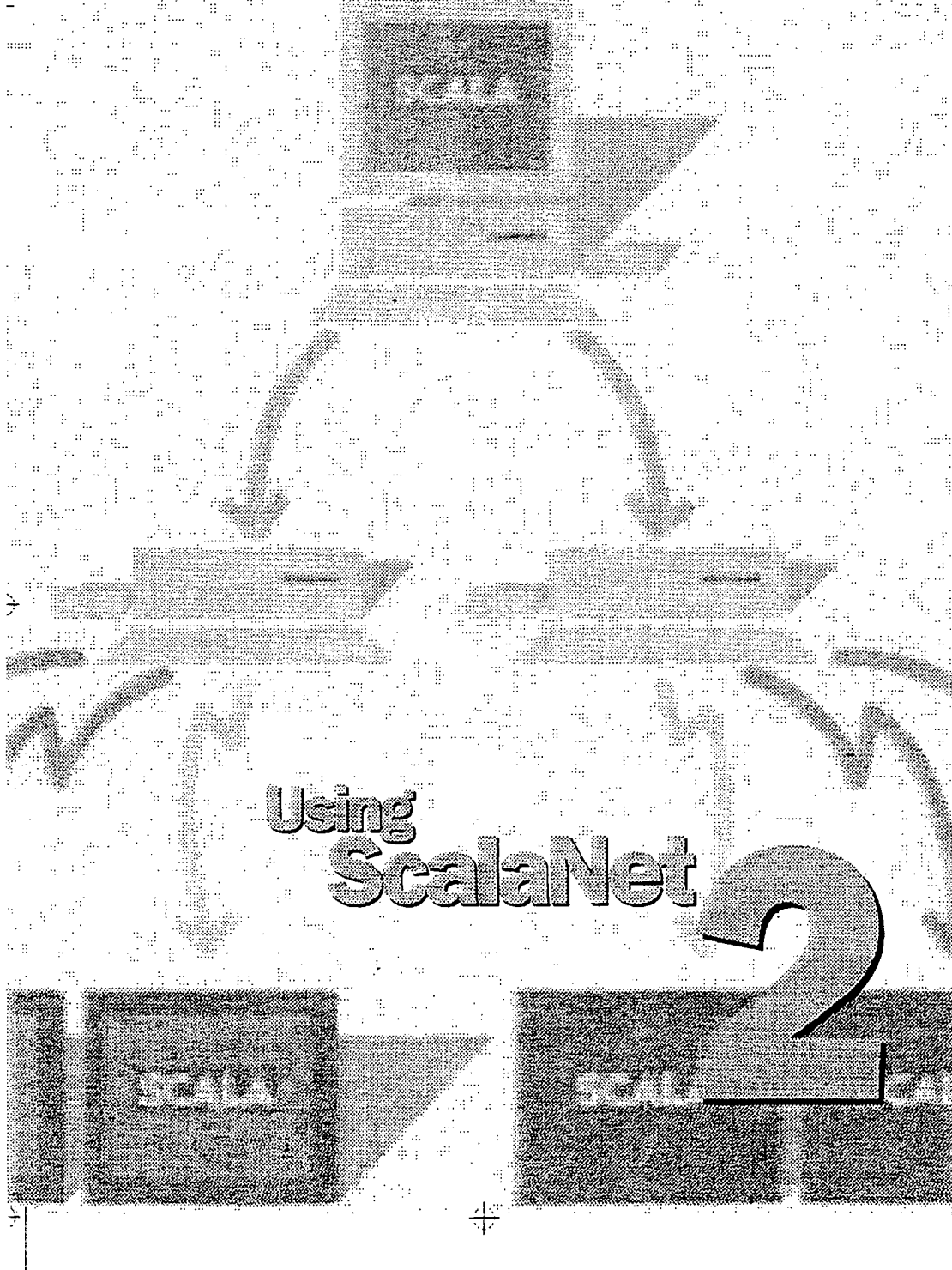
After creating the lunch page, we could schedule it to run as an exact-scheduled event with an *In time* of 12:00:00, *Repeat?* on, a *repeat time* of 00:10:00, and an *Out Time* of 13:00:00. In other words, it would run every ten minutes during the noon hour.

1: Scheduling InfoChannel events

Scheduling with groups and sub-scripts

Notice a few things about such a schedule:

- No matter what your script is doing at 12:00, 12:10, 12:30, 12:40, and 12:50, it will be interrupted to run the lunch page.
- If the script is longer than 10 minutes, some part of the script will never be shown during the lunch hour.
- If the script is short (perhaps 90 seconds), most people won't keep watching long enough to see the lunch page at all, because they will think they've seen the entire production.
- If the lunch page takes a long time to run (say, 12 minutes), it will interrupt *itself*, except the last time, when it will run past the lunch hour.
- The page will not start running at 13:00. 12:50 is the last repeat because 13:00:00 is the scheduled Out time.



2: Using ScalaNet

In most Scala InfoChannel applications, you create your entire production—the text, graphics, pictures, motion, sound and interactive functions—on one computer, a Master Station. From a Master Station, you then control and coordinate the distribution of your productions to one or more remote Player Stations. This coordination takes place through Scala's communication system, ScalaNet.

Master and Player Stations

Typically, Master and Player Stations are somewhat different. Because you use a Master system for creating and editing productions, it is generally a high-end system, with a fast Pentium[®] and at least 16 MB of memory, and possibly a removable hard drive or other additional hardware to make work easier. It probably runs Windows 95, and has other software applications for image processing, etc. It must have a modem or other means of communicating with its Players.

A Player system, on the other hand, can be more modest, because all it has to do is run the finished production and communicate with the Master. It requires only enough processor speed and memory to play productions at an acceptable speed, and could run DOS rather than Windows. It would include a VGA to video converter, if the final output is to video monitors. A Player also needs to have a modem or other means of communicating with the Master.

What a Player Station does is run a script you've produced, 24 hours a day. It runs this script, the Player Script, in a loop, over and over again. Whenever necessary, you update the script or its contents by "remote control" through ScalaNet from an InfoChannel Master Station. The multitasking nature of Scala InfoChannel and ScalaNet allows a Player Station to continue running its script uninterrupted even while it is receiving updates and new commands. This eliminates costly and annoying downtime, and provides a tremendous labor savings, because it is unnecessary for someone to physically update each Player Station's software whenever the script needs to change.

2: Using ScalaNet

ScalaNet capabilities

If you choose to use only one computer at a time for everything (producing, editing and playback), you don't need to know about ScalaNet. For most InfoChannel customers, however, the Master/Player architecture that ScalaNet allows is the essence of what makes InfoChannel such a powerful and versatile tool.

ScalaNet capabilities

ScalaNet communicates across any distance, using various communications technologies. With the Master Station on your desk, you control a Player Station located on another continent as easily as one in the same room.

Almost anything that you can do within InfoChannel from the keyboard of your Master Station, you can do remotely through ScalaNet on your Player Station(s).

Using ScalaNet, you can:

- modify scripts and elements by uploading changes from the Master to a Player.
- retrieve directory listings, scripts, or individual files from a Player.
- delete unused files from a Player.
- perform other maintenance, such as setting a Player's clock or rebooting a Player.
- retrieve log information or other files from the Player.

Other than to fix hardware problems or to perform major upgrades, through ScalaNet you can completely maintain an unattended Player Station from your Master Station.

Automatic repeats, password protection, and a number of other tasks are also controlled from ScalaNet.

Grouping is a ScalaNet feature that makes working with larger InfoChannel multimedia distribution systems easy. ScalaNet allows you to define groups of related Player Stations. This makes it simple to

2: Using ScalaNet

Example: A barker channel

update multiple Players with a single job. An addition, more than one Master Station may be used to control Player Stations, allowing more than one person at a time to work with the same set of Player Stations.

Although someone else—a dealer or VAR (value-added reseller)—may have set up your InfoChannel multimedia distribution system, you should know something about how it works. In this chapter, we first discuss that background information, then give details on how to use ScalaNet from a Master Station to control a Player Station.

Let's first look at an example application that uses ScalaNet.

Example: A barker channel

A "barker channel" is a simple advertising or community bulletin board channel typically displayed on cable television systems. It has a variety of text and graphics pages that relay local information of interest to the cable subscriber, and may include paid advertising.

For the purpose of this example, imagine that you work for a cable television company and have decided to use InfoChannel to set up a barker channel on your cable system.

Your barker system is going to be running 24 hours a day. Thus, you need to have one system that is continually producing the video output for the barker channel (this system is the Player Station), and another system on which you can make updates and changes to the content (this is the Master Station). Clearly, you need a way to copy those changes from the Master to the Player without interruption of the video feed. You will use ScalaNet to do that.

Because your cable company hasn't used InfoChannel yet, you decide to start small with just one Master and one Player.

You work with an InfoChannel value-added reseller (VAR) who specializes in cable TV applications, and come to the conclusion that your needs can be met with relatively simple hardware. Since the Player won't be located physically close to the Master, you decide to connect them by modem.

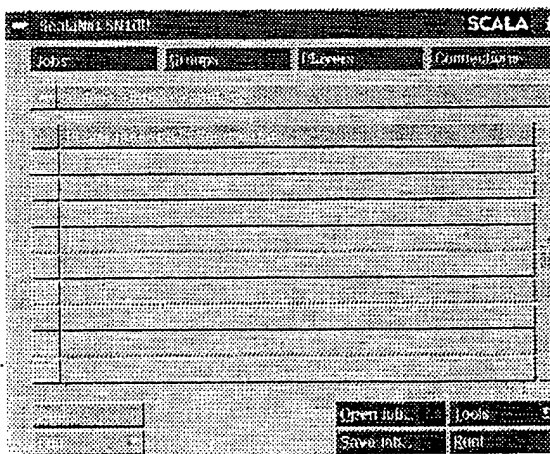
2: Using ScalaNet

Example: A barker channel

After settling on the details, you arrange for your VAR to set up the hardware and install the InfoChannel software for you, and to schedule you for a training class in using InfoChannel. After setting up the hardware, the VAR lets you know that the phone number for the Player Station is 555-4232 and has a password of 'fish'.

You decide not to wait for the training class. You've already put together a simple script you'd like the Player Station to run (you save it on the Master Station as BARKER.SCA), and you'd like to figure out how to copy it over. Using the HumanTouch interface, you work your way through the following steps.

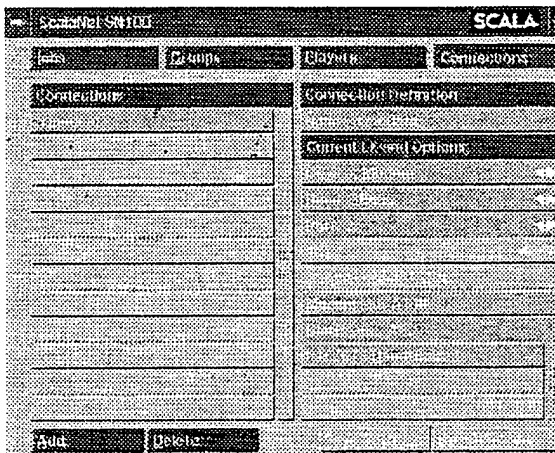
1. First, you start ScalaNet. On the InfoChannel Main menu, click the *Tools* pop-up button and choose *ScalaNet*. (You could also run ScalaNet as a separate program by double-clicking its icon or starting it from a command line.) You see the Jobs menu.



2: Using ScalaNet

Example: A barker channel

2. Next, you need to define a connection to use the modem. On the Jobs menu, click the *Connections* button. You see the *Connections* menu.

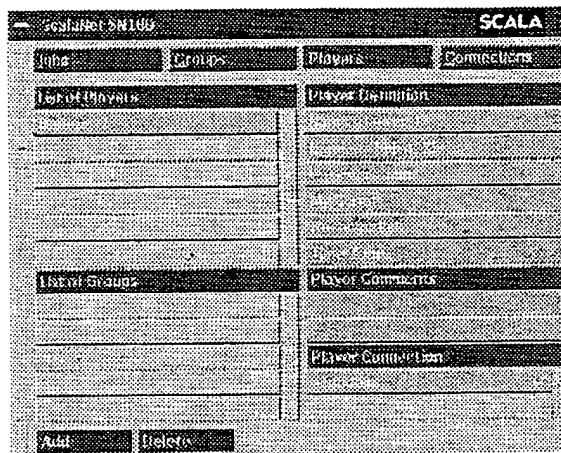


Because you have only one modem, you can name this connection "a modem". The recommended baud rate for use with ScalaNet and a 28.8 Kbps modem is 38400, so enter that. Your modem is attached to COM port 1, so leave everything else at their defaults.

2: Using ScalaNet

Example: A barker channel

3. Now you need to define a Player. Click the *Players* button. You see the *Players* menu.

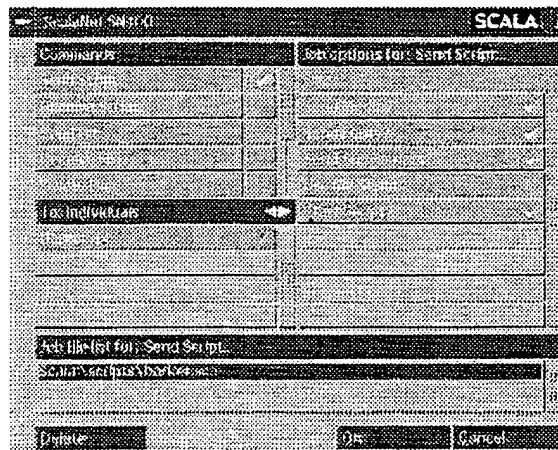


4. The Player will output video for channel 47, so name the Player "Barker 47". Also enter the phone number and password the VAR gave you for the Player. Finally, select (✓) the connection you'll use ("a modem", created in step 2) to connect to Barker 47.

2: Using ScalaNet

Example: A barker channel

5. Send your new script to the Player. To do this, you must create a job. Click *Jobs*. You see the Jobs menu. Click *Add*. You see the Add Jobs menu.



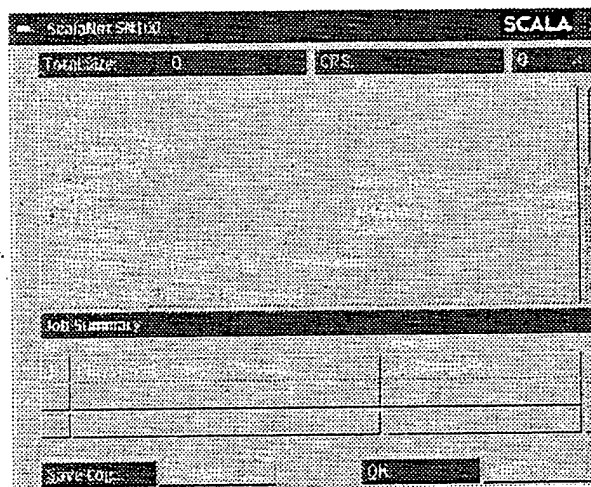
Select the Player "Barker 47" and choose the *Send Script* command. From the Send Script File menu, choose the script BARKER.SCA. (*Start Script* is set by default for *Send Script*, so your script will be run by the Player once it receives it.). You click *OK* to return to the Job menu.

6. Finally, you can run the Job. Click *Run* and you see the Job Control menu. Watch as ScalaNet dials out on your modem to the Player, which answers and connects (the VAR left the Player so that it is always running the InfoChannel Player software). You watch the progress as the files that make up your script are trans-

2: Using ScalaNet

Example: A barker channel

ferred to the Player. After the job is successfully completed, the menu would look similar to this:



The Player starts running the script you created, BARKER.SCA, and you're ready to start broadcasting on channel 47!

Topology and configuration

ScalaNet's flexibility allows you to configure an InfoChannel system in different ways depending on your needs. Traditional networks are made up of computers (or computers and terminals) and their connections. ScalaNet multimedia distribution systems are made up of Master Stations, Player Stations, the display units for the Player Stations, and their connections. The physical and logical layout of those connections is called the system's *topology*.

2: Using ScalaNet

Example: A barker channel

InfoChannel Note

Each Player Station requires an InfoChannel Player license; but any number of display units may be connected to each Player Station.

Before configuring your InfoChannel installation, someone analyzed what was required, and what topology would best meet these needs. In some cases, a Master Station and a single Player Station may be suitable to generate the display for a dozen display units, while in other situations a Player Station for each display might be controlled from a pool of Master Stations.

Regardless of the underlying physical connections, InfoChannel Master/Player communications may be configured in one of the following ways.

One to one

One Master Station is used for script editing and a Player Station for script playback. This allows you to edit the information without interrupting the display.

This simple setup might be used, for example, by a small cable TV operation to run a single barker channel.

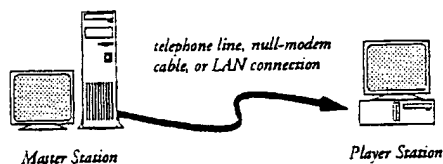


Fig. 1: One to one topology

One to many

One Master Station transmits information scripts to several Player Stations. You can send the same information to all Player Stations or you can transmit specific scripts to each site. With most communications technologies, a Master communicates with only one Player at a time, but by communicating with several Players, one after the other, this setup performs like a broadcast from the Master.

2: Using ScalaNet

Example: A barker channel

This topology would be used if our cable TV company has grown so that it has a number of barker channels.

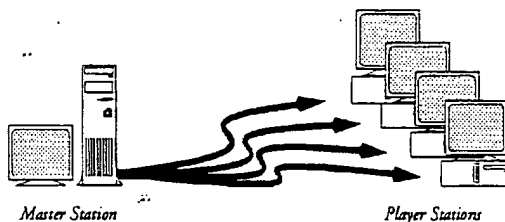


Fig. 2: One to many topology

Many to many

Several Master Stations can be used to transmit information scripts to several Player Stations. Each Master can send different information to the various receiving Players and the updating does not affect the rest of the script being played. Each Player can merge information from various Masters.

A large cable TV company with offices in different cities, with several barker channels in each city, would use this topology.

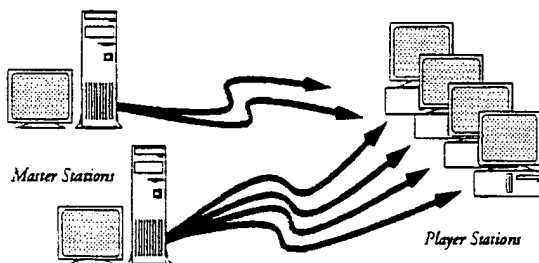


Fig. 3: Many to many topology

Physical configuration

ScalaNet currently supports several technologies for communications hardware, and Scala plans to support more in the future through additional Connection EXes.

2: Using ScalaNet

Example: A barker channel

Currently supported technologies include:

- Telephone (modem): Communication over long distances is most easily accomplished using standard telephone lines. ScalaNet's system of Intelligent File Transfer checks the Master and Player Stations when transmitting to avoid sending duplicate data. This reduces phone costs by not wasting time re-sending data already present on the receiver.
- Direct connect (null-modem): Where Player Stations are located near the Master Station, connecting serial ports directly—a *null-modem*—is inexpensive and easy. A nine-wire null-modem cable is required (both 9-pin and 25-pin serial ports can be used).
- Network (LAN): Local Area Networks are today's fastest and most reliable data communications tool. When Master and Player Stations are installed and configured on a LAN, they can share access to the same files on a network server. This does not involve a separate, InfoChannel-specific physical connection as modem and null-modem communication do, but works much as if there were such a connection. As explained in the section "Using a LAN," ScalaNet does not currently provide any special user interface on the Master side of a LAN connection.

Connection EXes

ScalaNet uses the Scala EX system for communication (see *The Scala EX system* on page 364 of chapter 16 for more information). This allows ScalaNet to be modular and extensible, so that it is easy to add support for new hardware.

Unlike other EXes, ScalaNet Connection EXes are kept in the SCALANET directory. To add a new ScalaNet EX to your system, you copy it to this directory. You use the ScalaNet Options menu to select which of the currently available Connection EXes to turn on or off. By default, all standard Connection EXes are on. Currently, Scala supplies three Connection EXes: *modem.ex*, *nullmodem.ex*, and *nil.ex*.

2: Using ScalaNet

Configuring a Player Station

Note that even when they are loaded (turned on), they are not actually used until you run a job that calls for them, so there is little point in turning any of them off other than a very small savings in memory usage. You should never need to delete any files in the SCALA\COMMEX directory.

You set the options for Connection EXes on the **Connections** menu of ScalaNet. To get there, click the *Connections* button on most ScalaNet menus. (For additional details, see the "Master configuration" section on page 39.)

Configuring modems and null-modems

Whether your serial ports use 9-pin or 25-pin connectors, ScalaNet uses readily available standard cables (or null-modem adapters) with at least 9-wire handshaking. Some serial cables and null-modems connect fewer wires, and those cables will not work with ScalaNet. Check your hardware against the diagrams on page 153 in Appendix H or consult your dealer for further information.

Configuring a Player Station

Your InfoChannel installation may already have been configured for you by the VAR or dealer who sold it to you, or by someone at your company. The following provides an overview of the configuration of Player Stations so that you can understand how a pre-configured installation works. If you need to configure a new InfoChannel installation, or for more detailed information on configuring Players, see the *InfoChannel Installation and Setup Guide*.

Note that Players perform best running under plain DOS rather than in a DOS window under Windows or OS/2.

Running the Player Script and other scripts

Normally, a Player runs a single script over and over in a loop. This is the production you created for the Player, and is called the *Player Script*. Although running this script is the Player's primary task, it is not the only script the Player can ever run. You can update the Player

2: Using ScalaNet

Configuring a Player Station

Script at any time with new backgrounds, text, clips, and sounds, or send it an entirely new script to become the Player Script.

When a Master sends a new script to a Player, it may send it with the option *Start Script* turned on. Sending a script with this option is the standard way for a Master to change what the Player is displaying. It causes the Player to load the received Script and run it. It also causes the Player to record the name of the script in the ICSTART.SCA file so that the next time the Player is started (if the Player is rebooted, for example), it runs the same script. A script sent with *Start Script* on becomes the Player Script.

Once a Player starts running a script, it continues running that script repeatedly. Only an error or a new *Send Script* command with the *Start Script* option turned on will change what script the Player runs.

Special scripts on the Player

Players by default have two additional scripts. The first, SYSTEM\READY.SCA, is what runs when the Player is first installed, before the Player receives its first *Start Script* command. It shows a background image that indicates that InfoChannel is running and ready to receive information.

There is another script included with InfoChannel, BACKUP.SCA, which runs if the main script cannot be run for some reason. It shows a standard color-bar test pattern so that the screen does not go completely blank if there is a problem. This alternative script should never be modified, because it exists only in case something goes wrong. If, for some reason, the alternative script cannot run either, the Player displays a black screen. In either of these cases, the Master can still communicate with the Player via ScalaNet, so by updating scripts you can remotely get the Player correctly running its Player Script again.

Passwords

When using a physically secure connection like a null-modem, passwords are unnecessary. Because it is a public medium, however, modem communication over phone lines is inherently subject to unauthorized access. The simplest way to prevent unauthorized individuals from interfering with your InfoChannel stations is to use pass-

2: Using ScalaNet Configuring a Player Station

words. If your InfoChannel installation requires password protection, see the "Security notes" section on page 65. A password, if any, is set on the Player during installation and is stored in SCALA\CONFIG\COMMGR.SCA.

Setting up an InfoChannel Player

The ICPlayer Installation program enables you to set up an InfoChannel Player automatically through a graphical user interface. The following information on how to manually edit the required files shows what the Installation program is doing for you and how an InfoChannel Player works.

1. Load the file SCALA\CONFIG\COMMGR.SCA into a text editor or word processing program that can save plain text files.
2. Locate the COMEX lines, which specify which Connection EX the Player will use. For example, the COMEX line for a null-modem connection would be:

COMEX ("nulmodem.ex");
3. One and only one connection EX must be active on a Player, so "comment out" (disable) the COMEX lines you will not be using. (Commented lines have two slashes (//) at the beginning of a line, which signals InfoChannel to ignore the line.) You do not need to change the time zone and daylight savings time fields, as those are all controlled only from the Master(s).
4. If you are using passwords, add the password for this Player between the quotes in the COMM.Password = "x" line. Note that passwords are case-sensitive and may include non-alphabetic characters such as numbers and punctuation. See the section "Security notes" at the end of this chapter for more information on passwords.
5. Save the file (be sure to save it as a plain text file).

Depending on which Connection EX you left uncommented in step 3, you may need to edit SCALA\CONFIG\NULMODEM.SCA or

2: Using ScalaNet

Configuring a Player Station

SCALA\CONFIG\MODEM.SCA (the Nil Connection EX doesn't have an associated configuration file).

If the Player uses either the Modem or the Nullmodem EX:

6. Load the appropriate file (MODEM.SCA or NULMODEM.SCA) into a text editor.
7. To set the connection speed, find the lines that begin with either

`modem.baud=`

or

`nullmodem.baud=`

For null-modems, the recommended speed setting is 57600, and must be identical to the speed set on the Master Station. (Using 115200 baud might not be reliable on all PCs, depending on the Player's CPU, serial hardware and other factors.) For 14.4 Kbps modems, the speed should be 19200, and for 28.8 Kbps modems, it should be 38400.

8. To set the serial flow control ("handshaking") method, find the lines that begin

`modem.rtscts=`

or

`nullmodem.rtscts=`

These should be set to on so that RTS/CTS (hardware handshaking) is used. The only time that these should ever be set to off is in an emergency situation in which you are forced to use a serial cable that does not allow RTS/CTS. See the cable diagrams on page 153 for proper serial cable wiring.

2: Using ScalaNet

Using InfoChannel on a LAN

9. To select the proper COM port (1-4 only), find the lines that begin

```
modem.comport=
```

or

```
nul modem.comport=
```

These should be set to 1, 2, 3, or 4, depending on the COM port the Player will use.

10. Now you are ready to run the Player. Enter the following lines at the DOS prompt, or put them in AUTOEXEC.BAT, replacing c:\scala with the path for the Scala directory on your Player, if necessary:

```
CD c:\scala
```

```
RunIC
```

```
autoexec
```

RunIC is what runs the InfoChannel software, starting the Player Script. The autoexec command causes the Player to automatically re-run AUTOEXEC.BAT if it should exit the Player Script for any reason. This ensures that the Player software is always running as long as the Player computer is turned on.

11. Save the MODEM.SCA or NULMODEM.SCA script file.

Using InfoChannel on a LAN

If your InfoChannel installation incorporates a LAN (local area network), you can use the LAN facilities for many of the operations for which ScalaNet is necessary when using modems. You use the LAN's networking software, such as Novell NetWare™, File and printer sharing for Microsoft Networks, or OS/2 Warp Connect, to create shared network drives. You then can put the scripts and data for your productions on these shared drives to take advantage of ScalaNet on your LAN.

2: Using ScalaNet

Using InfoChannel on a LAN

Configuring a Player on a LAN

To configure a Player Station for LAN use:

1. Be sure the Player is set up to monitor incoming communications with a Connection EX. (Use the *nil* EX if the Player is connected only through the LAN, and not by modem or null-modem.)
2. Load the file COMMGR.SCA into a text editor, and find the line `COMM.NetWatch = Off`. Change `Off` to `On`.
3. Save COMMGR.SCA.

The Player will both monitor the time/date stamp of the Player Script and listen for incoming connections on the Connection EX you configured, so you can still have complete control of the Player from a Master that is not on the LAN. Note that if you do not configure the Player with any Connection EX, it will not monitor the LAN.

When configured for LAN use, while your Player Stations run their scripts, they periodically check to see if their Player Script files stored on the LAN have changed. If the Player detects a change, it loads and runs the updated script file as if for the first time (from the first page). This is different from how a Player behaves when it receives an update via ScalaNet. With a ScalaNet update, even if you send the files for an update, the running script continues to run unchanged until a *Start Script* command is received.

InfoChannel Note

The Player software considers a script changed if the timestamp or file size of the Player Script file has changed, and does not examine any of the other elements of the script.

Thus, to make changes from a Master Station on a LAN, you do not need to use ScalaNet at all. You simply make changes to the script and save that file to the shared drive under the file name used by the Player Script. Note that to save all media and fonts with a script, you can use the *Copy to Dir* feature described on page 88 of chapter 3 in the "Producing" guide.

To perform remote maintenance, set the time, or perform other functions, you can use the administration utilities provided with your LAN rather than ScalaNet. Alternatively, you can use a modem or null-

2: Using ScalaNet

Configuring a Master Station

modem connection with ScalaNet for these functions, instead of or in addition to using the LAN.

Configuring a Master Station

Most configuration of the Master is done from within ScalaNet rather than by the installer program. You can easily add new connections or Players to your configuration at any time.

You can start ScalaNet on your Master Station in one of three ways:

From the InfoChannel Main menu, go to ScalaNet by clicking *Tools* and choosing *ScalaNet*.

or

Start it from Windows or OS/2 desktop by double-clicking the ScalaNet icon.

or

Start ScalaNet directly from DOS by entering

```
cd \scala  
scalanet
```

at the DOS prompt.

In any case, you first see the ScalaNet Jobs menu.

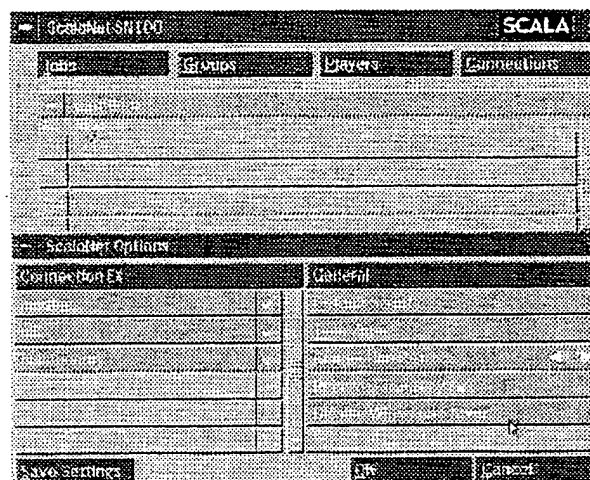
Setting options for the Master Station

ScalaNet's Options menu is where you do most Master Station configuration tasks. From the ScalaNet Jobs menu, click *Tools* and choose

2: Using ScalaNet

Configuring a Master Station

Options to open the ScalaNet Options menu. It opens on top of the Jobs menu as shown here:



Activating Connection EXes

To make sure that all of the Connection EXes you need to use are available and active, click on the right end of the *modem*, *nil*, or *nul-modem* buttons so that the EX is selected (✓).

Setting a default job script

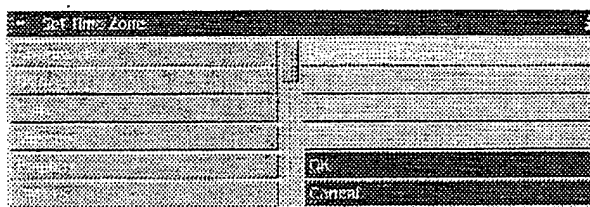
By default, ScalaNet starts without a ScalaNet job script loaded. You may find, however, that you regularly perform the same activities with ScalaNet and usually want to start with the same ScalaNet script. If you have a ScalaNet script currently loaded, turning the *Default Script?* option on (✓) makes the currently loaded script the default script. It will then be loaded whenever you enter ScalaNet. If you had previously set another script as the default, turning this option on changes the default to the current script.

2: Using ScalaNet

Configuring a Master Station

Setting a time zone

If your Masters and Players are in different time zones, click the *Time Zone* button to open the Set Time Zone menu.



Click on an entry in the scrolling list to set the time zone in which the Master is. A separate option lets you specify whether the Master is currently under Daylight Savings time. (This seasonal one-hour offset from normal time has other names outside the US.) Don't set a time zone on your Master unless you are going to set it for each of your Players (see the next section, "Connections").

In the Set Time Zone menu, you select from a scrolling list of major cities included in each time zone. Select (✓) the city which matches your time zone. Also select whether Daylight Savings time is currently in use in your area. You will have to manually change this in the spring and autumn. The default is to not use time zones. You can stop using time zones by clicking again on the selected zone. When you have finished in the Set Time Zone menu, click *OK* to accept your changes, or on *Cancel* to revert to the previous settings.

Controlling connection attempts

If an attempted connection fails, for example because the phone line is busy, ScalaNet immediately tries to make the connection again. Use the *Connection Retries* value control to set the number of times ScalaNet retries a connection before failing. The default is 3.

Receiving a Player directory listing

The *Receive Directory* option, which downloads a directory listing from the Player, is on by default in the Add Jobs menu. If you do not want it to be on by default, turn off the *Receive Directory?* option here in the

2: Using ScalaNet

Configuring a Master Station

ScalaNet Options menu. See the section on *Receive Directory* on page 55 for more information.

Disabling wipes

Although the InfoChannel Player software can continue to run a script while communicating with the Master, communications can disrupt the smoothness of wipes on slower Player hardware. One way to avoid this disruption is to select the *Disable Wipes on Player?* option. This option causes all wipes that occur in the Player script during ScalaNet communication to be temporarily rendered as Cut wipes.

Using and saving your Options settings

To use the option settings you have made on the ScalaNet Options menu and return to the Jobs menu, click *OK*. If you want to save these settings so that they apply to future ScalaNet sessions, click *Save Settings*. The changed options settings are stored in the COMMGR.SCA script. Click *Cancel* to return to the Jobs menu, discarding any changes you made.

Defining Connections

A *connection* in ScalaNet is a set of parameters, such as device type, baud rate, COM port number, etc. that you can name and store for use as needed. (It does not include a modem phone number—that is part of a Player definition.)

You need at least one Connection set up in order to communicate with your Players. If your dealer or VAR hasn't already set up the connections, this will be your first configuration task on the Master.

Being able to define a connection that is distinct from the Player definition is increasingly useful as your InfoChannel network grows. You can define any number of connections; there is no limit. You must have at least one for each device you use, and it is likely that, as the number of Players increases, you will have more than one connection for each modem or null-modem.

This is because the more Players you have to communicate with, the more likely that there will be different needs among them—they will

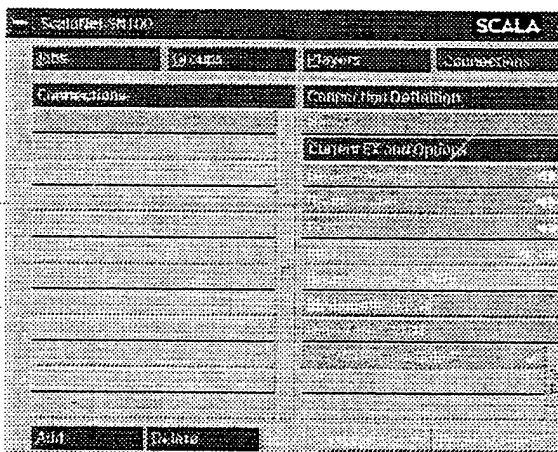
2: Using ScalaNet

Configuring a Master Station

use different modems with different capabilities, and have various special communication requirements. Having defined various connections means that you don't have to set up for each Player individually—you just pick from your "library" of connections.

1. Click the *Connections* button on any ScalaNet menu to open the *Connections* menu.

"<New Connection>"
appears here
after you click *Add*



2. Click *Add*. You see <New Connection> both under the *Connections* list on the left and on the *Name:* button under the *Connection Definition* on the right.
3. Change the text of the *Name:* button to a name you'd like to use for this connection. The name can be anything you want, up to 80 characters, but it must be unique on this Master Station.

If you have only one modem or null-modem connection, you might use the name of the device as the name of the connection. Otherwise, a name specifying the brand of modem or the serial port number, such as "modem com1", will provide better identification.

2: Using ScalaNet

Configuring a Master Station

4. Use the *Device:* selector under *Current EX and Options* to select a Connection EX. Choices include *Modem*, *NullModem* and *Nil*, assuming none of these has been turned off in the ScalaNet Options menu. Usually you use *Modem* or *NullModem*. The *Nil* device allows you to create a dummy connection that does not connect to any Player but is useful for experimentation with ScalaNet and for demonstration purposes. It is also needed for communicating with a Player that is accessed only through a LAN.
5. In the *Dial Prefix:* text box you enter a string that is sent to the Master's modem immediately before what is specified in *Number:*. The default is "ATDT", which signals the modem to open a touch-tone phone line in preparation for dialing the number. Your modem's documentation details the various possible options that this string might contain. You would also append any prefix—such as "9"—that your local phone system requires to get an outside line if the Player is external to that system.
6. Set the *Modem Init:* parameter, if necessary, to prepare the modem for transmission. This string can be used to configure the modem in any way required. (A modem's documentation lists possible configuration codes and their uses.) The default is "AT &F", which loads the modem's factory default settings.
7. Set the *Port*, *IRQ*, and *Baud:* parameters under *Current EX and Options* as needed for your device. Use the *Port:* selector to indicate which COM port (1, 2, 3, or 4) the device is using; this also sets default interrupt request (IRQ) and address values. However, you can set *Port:* to *IRQ/Address* if you need to specify a custom IRQ and address for special serial cards. Note that for a null-modem the *Baud:* value must exactly match that of the Player. For a modem, try the defaults for the other options, or consult the documentation that came with your modem for proper settings.
8. Click *RTS/CTS Handshake?* to turn hardware flow control on and off. Normally this option should be left on.

2: Using ScalaNet

Configuring a Master Station

- Click *Save Settings* to save the configuration for this connection. Note that clicking on any other connection, choosing *Add*, or going to another ScalaNet menu all have the same effect as *Save Settings*. You can revert to the last saved configuration for the current connection by clicking *Reset Settings*.

Players

Once you have configured the Master Station itself, you must then configure ScalaNet on the Master with information about each Player so that they can communicate.

- From any ScalaNet menu, click the *Players* button at the top of the screen. You see the Players menu.

"<New Player>" appears
here after you click Add

- If you have already set up some Players on this Master Station, they are listed and the first one is selected. The *Player Definition*, *Player Comments*, and *Player Connection* fields reflect the selected Player.
- Click *Add*. You see <New Player> both under *List of Players* on the left and on the *Name* button under the *Player Definition* on the right.

2: Using ScalaNet

Configuring a Master Station

3. Change the text of the *Name:* button to a name you'd like to use for this Player and edit the other parameters as explained below.

Any changes you make to the *Player Definition* are automatically saved when you select another Player, *Add* another Player or move to another menu.

Name

You can name your Players anything you like. This name is used only on your Master Station(s). If you have multiple Master Stations, you should keep the names of Players consistent on each Master (each Player should have only one name) This is not an absolute requirement, but should be treated as such to prevent confusion.

Number

The *Number:* text box is where you enter the number to be dialed for modem connection to this Player. For nil and null-modem connections it is unnecessary and is ignored if present. You may include anything in this string that your modem understands or that your telephone system requires. For example, in some business phone systems, a number sign (#) might produce a pause to get a second dial tone. Spaces and hyphens (-) to make the number more readable are accepted and ignored.

Password

For Players configured to use a password, you enter the password for that Player in the *Password:* text box. If the Player requires a password and the Master doesn't have the password correctly set here, the Master won't be allowed to connect to the Player. Note that passwords are case-sensitive and may include non-alphabetic characters such as numbers and punctuation. For more information on passwords, see "*Security notes*" at the end of this chapter.

My Script

When you send or receive a script, you can either specify the file name explicitly, or you can turn on the *Use My Script?* option. In the *My Script:* text box you can optionally specify the name of a script. This associates the script name—normally that of the Player Script—with the current Player. Thus it is a shortcut to refer the script you usually

2: Using ScalaNet

Configuring a Master Station

update on this Player: you just select *Use My Script?* for the *Send Script* or *Receive Script* commands, without having to know (and specify) the actual name. This is most useful when you use the Groups feature (see page 58). The script you specify here must be located in the Scripts directory.

Receiver Init

Receiver Init: is an optional parameter similar to *Modem Init:* on the *Connections* menu. It is a string sent to the Player modem *after* a connect, but *before* any ScalaNet initialization or data transfer takes place. This could be necessary, for example, when the phone number you dial for a Player is actually a special piece of equipment that selects from among a pool of modems to connect with a specific Player. Unless you will be connecting to such specialized hardware, you can safely ignore this option.

Time Zone

If you use the Master to set the time on Player Stations, and the Players are not all in the same time zone as the Master, then you must set the time zones on each machine (both Masters and Players). You must also manually select whether each Player is in Daylight Savings time (and manually change that setting when the Daylight Saving time status changes).

To set the time zone for this Player, click the *Time Zone* button to open the *Set Time Zone* menu. This menu works the same for setting a Player's time zone as it does for setting the time zone of the Master (see "*Setting a time zone*" on page 41).

To change to or from daylight savings time, click (✓) the appropriate buttons in the *Time Zone* menu.

Player Comments

You can use these text boxes to store brief information or important reminders about this Player, such as the name and phone number of the person to call if there is a problem with the hardware.

2: Using ScalaNet

Jobs

Player Connection

This is a list of the connections you defined in the **Connections** menu (see page 42 for details). You must select one for the current Player. The connection you choose must be appropriate for whatever hardware the Player will use.

List of Groups

This displays the list of groups (if any) that this Player belongs to. It is empty when you first add a new Player. If you use the **Groups** menu (described on page 58) to include some Players in some groups, then the next time you visit the **Players** menu you can see each of the groups any particular Player belongs to.

Jobs

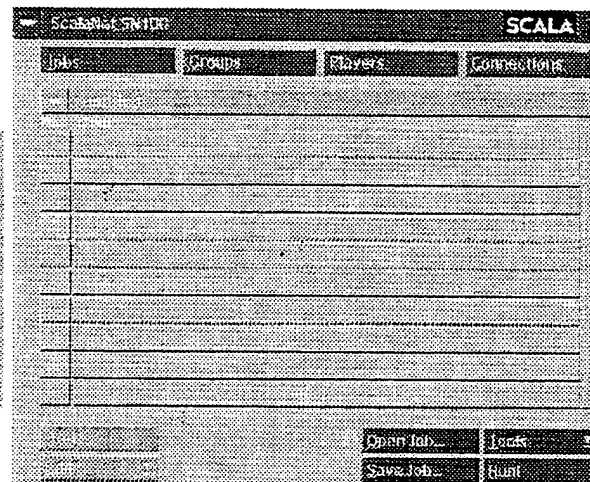
ScalaNet always works in a "batch" mode, rather than interactively. This means that you first create a list of tasks—in InfoChannel, called a *job*—and then execute the job. All ScalaNet communication is centered around a job on the Master Station. A ScalaNet job is really an InfoChannel script that specifies what communications are to take place. You create the jobs on the Master in the same way that you

2: Using ScalaNet

Jobs

author any other script, except that you do it from within the ScalaNet Jobs menu rather than from within the InfoChannel Main menu.

rows contain ScalaNet jobs



The Jobs menu is the primary ScalaNet menu. In the Jobs menu, you can add, delete or rearrange jobs in a ScalaNet job script, as well as open, save, and run ScalaNet scripts.

Each job you add to the list on the Jobs menu consists of one or more commands, all of which are directed to the same Player or Players. You may notice that this menu is similar to the InfoChannel Main menu in several ways:

- You can drag jobs up and down in the list to change their order
- The *No.* column shows the job number, and allows you to change the job name
- The name of the ScalaNet job script is shown in the script title bar
- The buttons at the bottom of the menu (*Add*, *Edit*, etc.) function in the same way as their Main menu counterparts

2: Using ScalaNet

Jobs

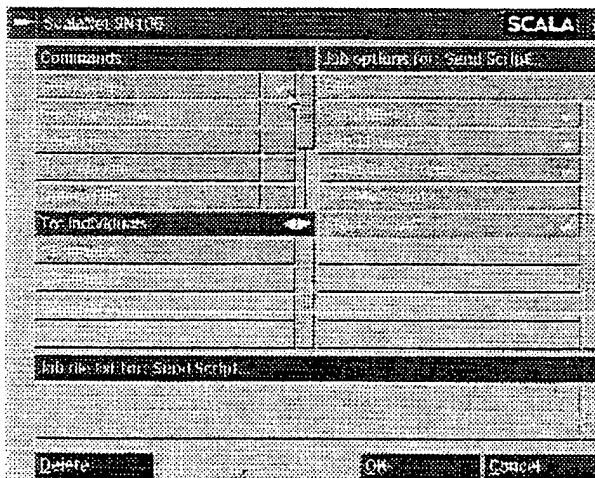
InfoChannel Note

Unlike the InfoChannel Main menu, the ScalaNet Jobs menu does not allow more than one script to be open at a time, or make it possible to turn a page off.

So from the Jobs menu, you can click and drag and use the *Edit* pop-up menu much the same way as you would on the InfoChannel Main menu. You can add new jobs, edit jobs, or change the order of jobs. You can also save and open job scripts so that you can perform common tasks without having to create the jobs again each time.

Creating a new job

To create a new job, click *Add* in the Jobs menu. You see the Add Jobs menu.



On the Add Jobs menu, you select from lists the commands you wish to perform, the Player(s) on which to perform them, and the options for those commands. Options include the names of files or scripts to act on, whether to transfer scripts complete with fonts and media, and so on. You can also use this menu to edit jobs you had previously created and stored.

2: Using ScalaNet

Jobs

Specifying the Player(s) for this job

First select the individual Player(s) and/or group(s) you wish to send your command(s) to, using the list beneath the *To:* selector. Use the *To:* selector itself to toggle between a view of individual Players and of groups. If you want to send a different set of commands to particular Players or groups, you need to create another job for each set.

Specifying commands

You can select one or more commands from the list by clicking on its button. Clicking on the main part of the button selects the command, so that you can see its options. Clicking the right end actually turns the command on or off. By default, the *Receive Directory* command is on (✓) for each new job (so that your copy of the Player's directory is kept current and you can always easily do maintenance, as discussed in the next section). When you select a command, the command name appears after *Job options for:* and *Job file list for:*, and those lists change to reflect the files and options you specify for that command.

You can select a command independently of turning the command on or off, allowing you to edit the options and files for that command. If you have not turned on a command, but indicate a file name for a command, ScalaNet turns on (✓) the command for you. Otherwise, as you create your job, make sure that you turn on each command as you are working with it, before you move on to editing the options for the next command.

InfoChannel Note

The order in which you select commands for a given job is irrelevant to their execution. ScalaNet performs them in a sensible order; a file marked both to be received and deleted, for example, is first received.

Note that you shouldn't send or receive scripts as files or vice-versa. If you try to send a non-script file as a script, you will get an error when the job is run. If you try to send a script as a file, you will not be able to include the sub-scripts, graphics, and other elements.

ScalaNet initiates separate connections for each job. Hence, you will usually want all of your commands to a given Player to be on a single job. One exception would be when some jobs are common to many Players, while others are unique to some individual Players.

2: Using ScalaNet

Jobs

Send Script

Select this command to send a script to a Player. Most often, this will be an updated Player Script. When you click the *Files:* button, you see the File menu. Here you select the script on the Master Stations that you want to send. (See chapter 4, "Using the File menu" in the "Producing" guide for details.) You can select more than one script to send. The destination location is Scala:\Scripts if a Home: directory has not been defined, or Home:\Scripts if it has.

When you've selected one or more scripts and clicked *OK*, you see the Add Job menu again. At the bottom of this menu, under the heading *Job file list for:*, you see the file name(s) of the script(s) you just selected. If you make a mistake or need to edit a job's file list, use the *Delete* button to remove the selected entry from this list of file names.

If you wish, you can send more than one script by repeating the above process.

Aside from *Files:*, the *Send Script* command has several job options:

Send Media?

On by default, *Send Media?* sends the files for any clips, backgrounds, sound sounds, etc., along with the file for the script. Recall that a script is actually composed of many files, and that the script itself is only a text file containing commands and references to external media. Turn this option off (no ✓) if you don't need to send the media files with the script files. You would do so if, for example, there are no media changes in the new script, only text changes, or if all the new media are already accessible to the Player.

Send Fonts?

On by default, *Send Fonts?* sends the files for any fonts used by your script. See the description for *Send Media?*, above.

Send Sub-scripts?

On by default, *Send Sub-scripts?* sends the files for sub-scripts along with the file for the main script. If *Send Media?* and *Send Fonts?* are on, the media and fonts used by the sub-scripts will also be sent, otherwise they will not.

2: Using ScalaNet

Jobs

Use My Script?

Turn on *Use My Script?* to use the file name you set up for *My Script* for the Players on the **Players** menu as the script to send. Otherwise, you would use the *Files:* button to select a script to send using the File menu. If you turn on *Use My Script?* and also specify a script with the *Files:* button, both scripts are sent.

Start Script?

On by default, the *Start Script?* option causes the Player to take several special actions once the script is received. First, it checks to ensure that the script is valid (it verifies that all media and sub-script files are available). Then, it saves the name of the script so that it becomes the Player Script. Finally, it terminates the currently running script and starts running the new (just sent) script.

Receive Script

You use *Receive Script* to retrieve a script from a Player. It has the all same job options as *Send Script* except *Start Script?*. The *Files:* button works differently when receiving a script, however. It is not possible to use the File menu to choose a file from a remote system, so instead you see a simple dialog. You must enter the entire path to the script on the Player. If the path you give contains directories that do not exist on the Player, those directories are created for you. Receiving a script overwrites the copy of that script on the Master.

Retrieving a script from a Player is not often used, but it can occasionally be invaluable. If, for some reason, a script had to be edited directly on the Player system, you could then update the script on the Master Station(s) using this command.

Send File

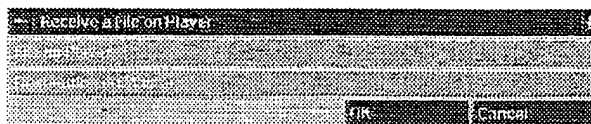
Send File works like *Send Script*, except that it has no options other than *Files:*. You can use this command to send any kind of script to a Player.

Receive File

Receive File is similar to *Receive Script*, but it has no options other than *Files:*. The Receive dialog looks like this:

2: Using ScalaNet

Jobs



It requires you also to specify a destination path for the received files on the Master. The path can be a full absolute path, as in C:\SCALA\CLIPS\DRAWINGS\BIRD.BMP, or it can be relative, as in SCALA\CLIPS\DRAWINGS\BIRD.BMP or HOME\AGFX\BIRD.BMP. If you include the file name in the destination path, the received file will have that name regardless of its name on the Player; in other words, this command can rename the file when it receives it. Doing so is important when retrieving log files (see the discussion of the Log EX in the next chapter). You can use *Receive File* to retrieve any type of file from a Player.

Delete File

This command lets you delete files on the Player Station. It has no options other than *Files*:

If you have obsolete scripts and media (sounds, graphics, etc.), you should periodically reclaim the hard drive space they consume. Note that deleting a script file deletes only that file, and not any of the media that the script may use.

InfoChannel Tip

You can and normally should use ScalaNet's Maintenance tool to receive and delete files from a Player. These options are useful here mainly as a way to edit jobs you created using the Maintenance tool. If you load a Maintenance job script, you can select *Receive File* or *Delete File* here to view and edit the job file list for that command.

For example, you might send an updated script to a Player Station once a day. You name each script to include the day's date so you can recognize older scripts. When you send your car-dealer sales script on May 10, you might then also delete CARS0509.SCA, CARS0508.SCA, etc.

There is no protection in ScalaNet for program files that are essential to InfoChannel or the rest of your system, so you must be careful when using *Delete File*. Never delete files whose function you do not understand. Delete only files you have created and/or that you know to be no longer necessary—outdated script, log, and media files, for example.

2: Using ScalaNet

Jobs

Receive Directory

Receive Directory retrieves a listing of everything in the Scala: directory of the Player. The listing is actually stored in a Player-specific file on the Master so that it is available for later commands. *Receive Directory* is enabled by default because referring to a current directory listing of the Player is generally the first step in most ScalaNet commands and maintenance tasks.

If the *Entire Disk?* option is selected (✓), this command returns a listing of all directories for the entire DOS drive that holds the Scala: directory. For example, if InfoChannel is installed to C:\SCALA, *Entire Disk?* (✓) returns a directory listing of the entire C: drive.

Set Clock

Set Clock sets a Player's real-time clock to be in sync with the Master's clock. If time zones have been defined for the Master and Player, the Player's clock is set to Player local time; otherwise the Player is set the same as the Master. This allows a script that displays the current time, for example, to display the correct local time. This command sets the time within a second, which should be sufficiently accurate for any InfoChannel application, but it should not be relied on for more precise synchronization.

Restart Player

If a Player does not seem to be operating correctly, and other efforts to remedy the situation have failed, you can use the *Restart Player* command to reset the Player software. On the Player side, it actually exits the Player software. This is on the assumption that it will be automatically restarted—that the Player Station is set up to automatically run the InfoChannel Player software when it boots.

Reboot DOS Player

The *Reboot DOS Player* option is similar to *Restart Player*, but should be used only on Players running DOS. It performs a warm boot, just as if someone had pressed Ctrl+Alt+Delete. If the Player is running under an operating system other than DOS, the reboot option may not perform as expected. Under Windows 95, for example, it would

2: Using ScalaNet

Jobs

close the DOS window the Player was running in, leaving the Player at the Windows 95 desktop screen.

Intelligent File Transfer

When you send a script update from a Master to a Player, you have control over which parts of a script are sent, and may choose not to send fonts, media, or other components, as described previously. But for each file that fits the transfer criteria, ScalaNet may or may not actually send it. ScalaNet checks the file size and date stamps between files that exist on both the Master and the Player. If they match, Scala doesn't transfer the file. This saves the time and expense of transferring files which haven't changed, while giving you the confidence that every part of a script that you have modified will be updated (unless you specifically instruct ScalaNet not to update some parts of a script).

One exception is the script itself. When you send a script to a Player, it is always sent, regardless of date stamps and file size.

While a Player is receiving updates, each file sent is received into a temporary file until ScalaNet has completely transferred that file. This ensures that an error during the communications process doesn't leave any files in a half-updated state.

ScalaNet Tip

To see how something would work without actually changing anything on a Player, create a test Player, and select the NII as your connection. Then you can create and run a new job without really being connected to a Player, or actually changing any files. Note that you cannot use Maintenance via a NII connection.

The time/date stamp on a file is sent along with the file, so that the Player will have the same time/date stamp for received files as the Master that sent it.

During ScalaNet communications, the Player Station continues to run its script. Scala's multitasking means that the display from the Player isn't interrupted for communications. After a Player receives all updates, it may run a newly updated script, depending on the commands sent by the Master.

A script to be sent with ScalaNet is validated at both ends. First, the Master side assures that the script doesn't refer to files which don't exist and that the script is otherwise valid. If the script doesn't pass these tests, it isn't sent. Once a script is transferred, the Player performs sim-

2: Using ScalaNet

Running ScalaNet scripts (jobs)

ilar checks so that a script is not run on the Player if the Player can't run it without obvious errors (again, such as missing files or syntax errors). With the most complex scripts, of course, there may be subtle errors that are not detected by these tests, and are only encountered during actual running of the script. For this reason, it is important to test scripts after updating Players with them, as well as to monitor Player log files to be sure of what the script actually did (see the "Log EX" section of chapter 3, "Additional InfoChannel EXes").

Exiting the Add Jobs menu

When you have completed editing your job on the Add Jobs menu, use the *OK* button to finalize the changes you've made. The job appears on the Jobs menu with a name consisting of the Player name and the first command in the job. (You can change this name if you want, by clicking on its *No.* column button.) If you don't want to add the job to the job list, click *Cancel* on the Add Jobs menu to discard your changes.

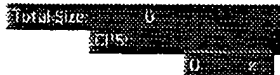
Running ScalaNet scripts (jobs)

A rectangular button with the word "Run" in a bold, sans-serif font.

Once you have created your ScalaNet job script, run it by clicking the *Run!* button on the *Jobs* menu.

Monitoring job progress

When a job script is running, you see the Job Control menu as shown on page 29. This is an informational screen that shows you the progress of the job, and allows you to halt the job.

A screenshot of the Job Control menu showing a table with columns for Total Size, CPS, and Percent Completed. The values are 0, 0, and 0 respectively.

Information provided includes *Total Size*: (the number of bytes sent and received for the job), *CPS*: (characters per second—the speed of communication during the job), and the percent of the job that has been completed (%).

Messages describing the progress of jobs is displayed in the center of the screen. For each connection to be made, information is displayed as the connection is established. For a modem connection, first the strings sent to the modem to initiate a call, then the modem's

2: Using ScalaNet Groups

responses are displayed. After a connection is made, information about each command sent is displayed. Any errors encountered are also reported in this area of the screen. The *Job Summary* at the bottom of the screen shows a less detailed version of the above display. The summary shows the status of all jobs including any pending jobs and jobs that failed. You can scroll either list back to see the status of earlier jobs.

The buttons at the bottom of the screen allows limited control of the job in progress, and different actions after the completion of a job. While a job is active, you can only watch or use the *Cancel* button to halt the job. After the job has finished (whether successfully or not) you have two options: *Save Log* and *Restart*.

Save Log



Choose *Save Log* to create a text file with a copy of the progress information that was displayed about the job just completed. Log files can be vital in keeping track of the exact status of your Players, especially in larger InfoChannel installations. Note, however, that the log files saved with this feature record only job operations. To keep a log of the events within your InfoChannel productions, you need to use the Log EX, described in the following chapter.

Restart



Restart becomes available only if there was a failure during a job. Clicking *Restart* re-runs any job script(s) that failed. In general, there is no harm in repeating an entire job if some part did not complete successfully.

When you are finished on the Job Control menu, you can return to the ScalaNet Jobs menu by clicking *OK*.

Groups

If you manage more than one Player, your job will be easier if you take advantage of ScalaNet's group features. Without the group features, you must enter separately each job that you wish to send to each

2: Using ScalaNet Groups

Player. With groups, you can treat several Players as one, sending them all the same script, for example, with just one command. You can define many different groups, each of which can contain any number of Players. You can even make Players belong to more than one group.

Note that the *My Script* field in the Players' definitions is particularly useful when using groups in ScalaNet jobs (see the "Jobs" section, below).

From other ScalaNet menus, when you click on the *Groups* button, you see the *Groups* menu.

ScalaNet SW 1.00																																																												
Jobs	Groups	Players	Connections																																																									
List of Groups		Group Definition																																																										
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> <th>Action</th> </tr> </thead> <tbody> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> </tbody> </table>		Name	Description	Action																															<table border="1"> <thead> <tr> <th colspan="2">Group Definition</th> </tr> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> </tr> </tbody> </table>		Group Definition		Name	Description																				
Name	Description	Action																																																										
Group Definition																																																												
Name	Description																																																											
Add		Delete																																																										

In the *Groups* menu, you use the left side to create, select or delete groups, and the right side to define the selected group (if any). If you have not yet defined any groups, Players, or connections, the menu has all fields blank, as in this illustration.

To create a new group, follow these steps:

1. Click *Add*. You see *<New Group>* in the *List of Groups* on the left, and also under *Group Definition* in the *Name* text box.

2: Using ScalaNet

Example: Multiple cable channels

2. Use the *Name:* button to change the name to the one you'd like to use to identify this group.
3. If you need to remember anything special about the group, enter something into the *Group Comments* text boxes. Otherwise, you can leave them blank. These comments are for your use only, and are ignored by ScalaNet.
4. Under the List of Players, select each Player that you would like to be a member of your new group, so that it is marked with a ✓.
5. The final option is rarely used, but if you wish to use the same device to connect to all of the Players in the group, you can use the *Group Connection* section to select that device. This might be useful, for example, if your Master has several modems, and you wish to connect to all of the Players in a particular group using a particular modem, regardless of which connection is specified in the individual Player definition.

In the Groups menu, any changes that you make to the Group Definition are automatically saved when you select another group or exit the Groups menu. You can delete a group by selecting it under *List of Groups* and clicking *Delete*.

Example: Multiple cable channels

Let's return to the cable TV example we started with. After successful trials of InfoChannel, your company decides to add more Players to handle other channels. This will be cost-effective because the company has multiple head-ends that mostly run the same programming.

Your nine-channel system has three head-ends: one for the local city, one for the surrounding suburban area, and one for a more distant city. Each head-end has:

- One channel that is a barker channel for your cable company—the channels all play the same simple slideshow-style production with information and phone numbers urging people to subscribe to additional pay channels.

2: Using ScalaNet

Example: Multiple cable channels

- An information channel that displays weather data and current events; in this case, the weather information is common to all the channels, but the current events for the distant city are different from the local urban and suburban channels.
- An advertising channel that runs a slideshow of advertisements for local businesses. Each of the three channels carries different ads.

In consulting your VAR, you decide that it is more convenient to use InfoChannel and send identical scripts to a Player at each head-end than to transmit the video from one head-end to another.

You therefore install three Players at each head-end. You could use the grouping, connection and My Script features of ScalaNet as follows:

Grouping the channels

With multiple head-end locations that all need the same content, you can create a group for each feed so that all the Players meant to display the same content are in the same group. Thus you could create the Barker group for the three barker channels; the Weather group for all three info-weather channels, and a Localevents group for the two local info-weather channels.

Although the advertising channels all run different programming, you might also create an AdChannel group in case you occasionally need to run an ad for a regional event like a large trade show on all three channels.

Using My Script

Because the scripts for the barker channels are all identical, they can all have the same name on each Player, and you don't need to set *Use My Script?*. When the barkers' Player Script needs to be updated, you would just send the same script to the Barker group with *Start Script?* on.

When updating the scripts on the other Players, however, the My Script option would be useful. Because the programming can differ from Player to Player, you would want to give the Player Script on each Player a different name, such as INFOCITY.SCA, INFO-

2: Using ScalaNet

Example: Multiple cable channels

SUB.SCA, and ADCITY2.SCA. You put these names in the *My Script*: text box (on the *Players* menu) of each *Player* definition on the Master, and maintain the current versions of each of these scripts on the Master Station.

Then, when you need to send new versions of each of these scripts to the *Players*, you don't need to create a separate job for each uniquely-named script. For each group, you just create a single *Send Script*: job with *Use My Script?* selected (✓), and ScalaNet will send each *Player* the script named in its *My Script*: definition.

Connections

There are many ways you might set up connections with the nine *Player* systems, depending mainly on the modem hardware you have. Some simpler examples include:

One connection – the simplest scheme is to have a single connection, which would communicate using a single modem in the Master. However, this would be possible only if all the *Players*' receiving modems had matching configurations: the same baud rate, initialization requirements, etc.

Connections at different speeds – You might have *Player* modems that have different maximum communication speeds. You would need a connection for each baud rate.

Modem and null-modem connections – If one of your head-ends is at the same site as the Master, you might use a null-modem to connect to it, and modem connections for the other sites.

Several Master modems – If you have more than one modem in the Master system—perhaps necessary if not all your *Players* modems are compatible with a single Master modem—you would need a separate connection for each.

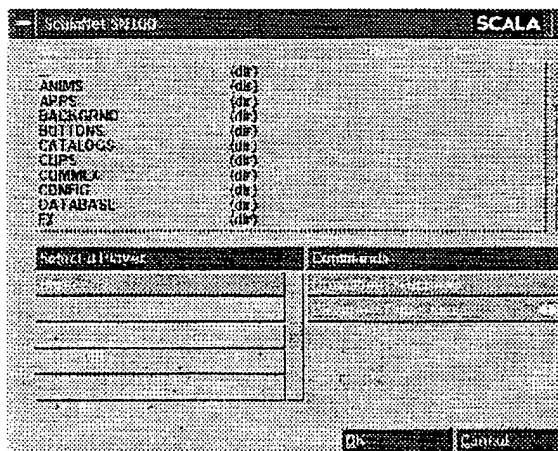
2: Using ScalaNet Player maintenance

Player maintenance

InfoChannel Players are normally located far from their Master Stations and are only accessible via modem. In some cases InfoChannel Players can be hundreds or thousands of miles away. There needs to be a way in which a InfoChannel Master can perform maintenance duties remotely without disturbing the playback that is going on in the foreground. While the normal ScalaNet job interface can be used for this maintenance, there is a simpler way. The ScalaNet Player Maintenance tool assists you in the creation of maintenance job scripts. It lets you download and browse through directory listings for remote Players, and easily mark files for retrieval or deletion.

The easiest way to understand the Maintenance tool is to try it.

1. From the ScalaNet Jobs menu, click *Tools* and choose *Maintenance*. You see the Maintenance menu.



2. Select a Player from the list at the lower left. This is the Player that your maintenance operations will affect until you select another Player or exit the Maintenance menu.

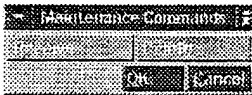
2: Using ScalaNet

Player maintenance

3. If this is a new Player, or a *Receive Directory?* job has never been performed on it, "Directory unavailable" is displayed. Click *Download Directory*. You see the Job Control menu, and can watch the progress as a connection to the Player is established and a directory listing is downloaded from the Player to your Master Station. (Note that this is normally not necessary, provided a job that included the *Receive Directory?* option has previously been run.)
4. Click *OK* to exit the Job Control menu. You see the Maintenance menu again. Now it displays the contents of the Scala: directory.
5. If you double-click on a directory name, you see the contents of that directory. If you double-click on a file, you see the Maintenance Commands dialog. Navigate through the directories to click on a file you'd like to download from the Player and/or to delete.
6. When you double-click an entry in the Files list, you see the Maintenance Commands dialog.

You may set the file to *Receive* and/or to *Delete* by selecting (✓) those options. Once you've set those options, click the *OK* button to accept your choices, or the *Cancel* button to discard them. (if you select both *Receive* and *Delete*, the file is transmitted before it is deleted from the Player.)
7. After marking all the files you want to act on, exit the Maintenance menu by clicking *OK*. You see the Jobs menu with a new job, which has been created by the Maintenance tool.
8. Run the new job by clicking the *Run!* button.

Note that the maintenance tool acts by default only on the Scala: directory of the Player, as determined by the *Directory Type:* selector. You can use the selector to operate instead on the *Home* directory (the directory in which all the scripts' media files are stored on the Player), or on the *Entire Disk* (the disk on which Scala is installed on the Player—the Maintenance tool cannot access other disks).



2: Using ScalaNet

Security notes

Remote reconfiguration

It is possible to remotely change the configuration of a Player. You can retrieve files from the Player's SCALA\CONFIG directory, modify those files, and upload those files back to the Player. This requires advanced knowledge of the configuration files, since the graphical tools for installing a Player are not available for use on the Master. Also be forewarned that this procedure can be risky, in that any mistakes you make could potentially leave the Player in an unusable state, requiring someone to visit the site to fix the problem at the Player unit itself.

Security notes

Any installation which provides remote access capabilities (especially via dial-up modem) must be concerned about security, thus you should consider the security implications of your Player Stations. ScalaNet includes password protection to minimize the risk of unauthorized remote access to Scala Players. The protocol used by Scala requires that if the Player has a password, the Master send an initial packet of information to the Player which includes, among other things, a correct password. Thus, as long as you set passwords, it should be quite difficult for an outsider to gain unauthorized remote access to a Player—it requires correctly guessing the password as well as either using an InfoChannel Master Station or software to emulate one.

There is currently little protection in ScalaNet against attempts to brute-force guess at a password. You should maintain a log and monitor it for repeated failed password entries (see chapter 3 of this guide, "*Additional InfoChannel EXes*"). As with any password, those you select for your Players should not be easy to guess—ideally, they should be at least six characters long, not be words in the dictionary and contain mixed case (passwords are case-sensitive), numbers or punctuation as well as letters.

Also note that any Scala Master or Player is subject to tampering by anyone with physical access to the PC hosting the Master or Player. Since passwords for each Player are generally stored on the Master,

2: Using ScalaNet

Exiting ScalaNet

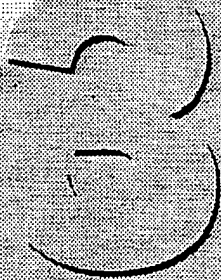
anyone with access to a Master can easily tamper with all the Players normally controlled by that Master. Depending on your security requirements, this may necessitate keeping both your Masters and Players in carefully controlled, locked facilities.

Exiting ScalaNet

When you exit ScalaNet, you see the InfoChannel Main menu again, or the DOS prompt or desktop you started from.



Additional
InfoChannel
EXES



3: Additional InfoChannel EXes

The Scala EX system provides a modular interface that makes it easy to add new features to InfoChannel. Such features include support for special hardware interfaces to external devices, support for additional file formats, and special functions that extend InfoChannel's flexibility.

Most EXes are kept in the SCALA\EX directory. To add a new InfoChannel EX to your system, you copy it to this directory. Use the InfoChannel Options menu to select which of the currently available EXes to turn on or off. (You should never need to delete files from the SCALA\EX directory. You can turn an EX off if you want to eliminate its column from the Main menu, or if you think it might be causing a problem with the system.)

ScalaNet Connection EXes are kept in the SCALA\COMMEX directory. To add a new Connection EX to your system, you copy it to this directory. You use the ScalaNet Options menu to select which of the currently available Connection EXes to turn on or off.

Two EXes provided with this release of InfoChannel are the Log EX and the Text File EX.

Log EX

The Log EX provides a mechanism to automatically monitor the activities of InfoChannel Player Stations. This is an essential tool to use in maintaining remote Players and providing a record of their activities. Logging capability is essential in a cable television installation, for example, where you need to know when and how many times advertisements were actually broadcast. Log files can also be a key tool in discovering the cause of any problems that might arise with a Player.

Log files

The Log EX keeps records by generating plain ASCII text files. It records entries for many actions by default, and you can specify addi-

3: Additional InfoChannel EXes

Log EX

ditional information to be recorded in these log files for individual script events.

The current log of a Player is always kept in SCALAMC.LOG. This file is copied to a backup log, SCALAMC.BCK at midnight every day. (If IC.BCK exists, it is overwritten.) Thus, two log files are always available. You generally don't need to worry about log files getting too large because they each contain only a single day's information.

Every five minutes that the Player is running, the IC.LOG file's time stamp is updated. This enables the InfoChannel Master to determine how long a Player has been operational in a 24-hour period.

How events are logged

Whenever a Player communicates with a Master (or loads an updated file from a LAN), information about the communications are recorded in the log. If you have specified a remark to be logged whenever some particular script event executes, that remark is recorded in the log when the event begins.

If the Player encounters any kind of error with communications or with running its script, it records information in the log about the error. Most Player Stations are used in environments where an audience sees everything the Player outputs, so it is not appropriate for Players to display information about errors. If something is wrong with your Player, you need to check the log files to find out what went wrong and when.

Reviewing log files

To review the log of a Player, you create a ScalaNet job to retrieve the file IC.LOG from that Player. You can use any text editor (such as the Windows Notepad) to review the log data. Note that if you wish to review the logs from several Players, you must rename each log as you retrieve it, otherwise each log retrieved would overwrite the previous one. This is easy to do by providing a new name in the *Receive File* dialog's *Destination on Master* text box.

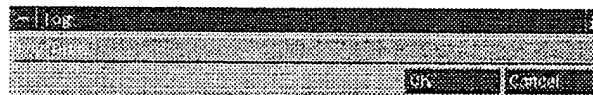
3: Additional InfoChannel EXes

Log EX

If you can't communicate with a Player for some reason, or it isn't executing a script properly, review the log for that Player. The log will usually contain information that can help you diagnose the problem. If you are responsible for remote Players, you should review logs from those Players on a daily basis to be sure that they are doing what they are supposed to be doing.

The Log menu

To have the Log EX record an entry when the Player runs a particular event (a page or an event on a page), edit the script on your Master Station. From the Main or List menus, click the button for the event in the Log column. You see the Log menu.



Enter the text to be recorded to the log, then click *OK*. The text can be up to 1000 characters, but normally need be only a few words. As with other InfoChannel menus, you can select and work with any number of events without leaving the Log menu. The text you enter also appears in the event's button in the Log column.

When you next send the script to the Player and it runs, the Player's log will include an entry with the text each time that event is run. If you no longer need to keep track of an event, simply delete all the text from its *Log Text* text box.

Typically, you might use log files to keep a record of how many times a page is run and to be able to verify that scheduled pages are running when they should.

One very powerful feature of log files is that you can include one or more embedded variables in the log text. When the text is written to the log file, the variable's current value at that time becomes part of the logged text. This allows you to gather and record new information with the log files. For example, you might use it collect statistics on the number of times per day that a given interactive button was selected by users.

3: Additional InfoChannel EXes

Text File EX

Text File EX

Typing text onto a page from the Design Text menu is easy and is sufficient for many productions. Sometimes, however, you need to work with large amounts of text, or text which comes from a remote source. Such situations make a different interface desirable. The TextFile EX gives you the capability to create a template page and have InfoChannel fill that in automatically with text from an external ASCII text file. It also provides for certain unique special effects.

The basic steps to use the Text File EX are:

1. Create a text file.
2. Create a script with one or more template pages.
3. Assign the text file to the template pages in the Text File EX menu.
4. Run the script.

As the script runs, the text file is displayed a certain number of lines at a time (you specify the number) against the background of the template page until the end of the text file.

Text files

The first step in setting up pages using the Text File EX is to create the text files you will use. These must be plain ASCII files, created outside of InfoChannel with a tool such as the Windows Notepad or the DOS EDIT command.

Each line that you want to appear on the page must be a separate line ending in a carriage return in the text file. Lines can be up to 1024 characters in length, but the practical maximum for general use is much shorter—approximately 100 characters, using the smallest possible font. You must ensure that the maximum line length in the file is not too long to be displayed on the page. Lines can be blank (just a carriage return). The text file can contain any number of lines.

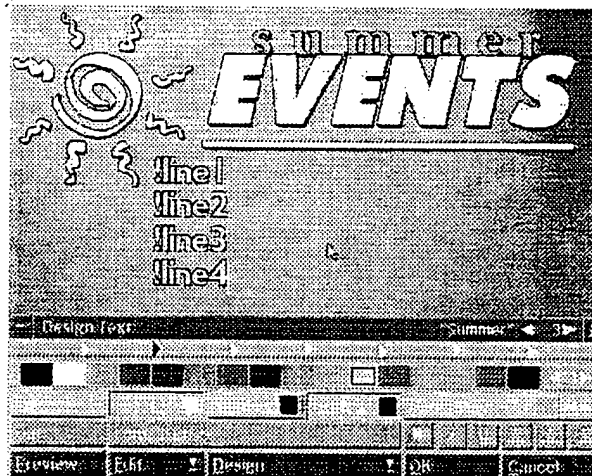
3: Additional InfoChannel EXes

Text File EX

Templates

A single template page specifies the page background and all other properties for the display of a text file of any length. The template page is a normal page, and you create it just as you create any other page in a script. Using the Design menus, you set the background, and any static text, clips and wipes to be used with the page.

You specify where each line of the text from the file will be placed by placing special embedded variables as text elements on the page. For the Text File EX, the variable text elements must be of the form !LINE1, !LINE2, !LINE3... !LINE99, as shown here:



Most templates will use much fewer than 99 lines on a page, but 99 is the maximum. (For more on embedded variables, see the section "Displaying variables" on page 164 of chapter 6.)

You can give the variable text elements any styles and attributes that normally belong to a text element—font, color, outline, shadow, wipe, etc. When the Text File EX substitutes the lines from the text file for the variables, they will have those styles.

3: Additional InfoChannel EXes

Text File EX

Text File menu

From the InfoChannel main menu, click the *TextFile* button for the template page you have created. You see the Text File menu.



Specifying the text file

In this menu, you specify the file name for the text file to associate with this template page, and the number of lines of text to display on each page.

When you click on the *File...* button, you see the File menu, which defaults to the SCALA\TEXT directory. You can also type the name of the file directly into the *File name:* button without using the File menu.

If the text file does not yet exist at the time you are creating the script, you must use the *File name:* button. When you enter a name here that does not exist, you see a dialog stating that an empty file must be created and asking if you want to proceed. If you click *Yes*, a zero-length file by that name is created for you as a placeholder. Sending this empty file to the Player can sometimes be useful; see the next section, "*Running the script*".

Use the *Maximum Lines on Page* value control to indicate the number of lines of text to be displayed on each template page. Usually, all template pages will have the same number of lines on a page, and this value should equal that number. If this value is not equal, only the number of lines specified by this value will be displayed on any page (regardless of how many !LINES are on the template). If the *Maximum Lines* value is greater than the number of !LINES on the template page, some lines in the text file will never be seen.

When you have finished, click *OK* to return to the Main menu and accept the changes you have made, or click *Cancel* to return to the Main menu without keeping any changes. If you later decide to make

3: Additional InfoChannel EXes

Text File EX

and in the Text File EX menu for the group, select the text file. When the script runs, each set of text lines appears over all five backgrounds before the next set appears.

Another useful effect you can create with the Text File EX is a scrolling banner-style message at the bottom of the display. Create a text file with a message in a single long line. Create a template page with !LINE1 in the lower right-hand corner of the screen, and assign the Crawl West fly-on wipe as the In wipe for this element. When the script runs, your message scrolls across the bottom of the screen until the end of the line in the text file is reached.

You could achieve this same effect by typing the message into a long text element in the Text menu, but this would be impractical, and would not allow for remote updating of the message.

